OWASP TOP 10

SIDDHARTHA NAIK
21BRS1056
VIT CHENNAI

A01:2021 – Broken Access Control

vulnerabilities in authentication mechanisms arise in one of two ways:

- The authentication mechanisms are weak because they fail to adequately protect against brute-force attacks.
- Logic flaws or poor coding in the implementation allow the authentication mechanisms to be bypassed entirely by an attacker. This is sometimes referred to as "broken authentication".

In many areas of web development, logic flaws will simply cause the website to behave unexpectedly, which may or may not be a security issue. However, as authentication is so critical to security, the likelihood that flawed authentication logic exposes the website to security issues is clearly elevated.

Lab: 2FA bypass using a brute-force attack

Some developers try to prevent OPT brute-forcing by logging a user out after certain number of retries but this can be hacked by using automated scripts, there is a lab to practice this vulnerability in port swigger academy.
I use macros to automatically enter the password for me
after that I use turbo intruder to brute-force the OTP

The code I used:

```
# Find more example scripts at https://github.com/PortSwigger/turbo-intruder/blob/master/resources/examples/default.py
def queueRequests(target, wordlists):
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=5,
                           requestsPerConnection=100,
                           pipeline=False,
                           engine=Engine.BURP
```

```
                            )

    for i in range(0, 10000):
        code = '{0:04}'.format(i)
        engine.queue(target.req,code.rstrip())

def handleResponse(req, interesting):
    if req.status==302:
        table.add(req)
```

A02:2021-Cryptographic Failures

shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.

Some examples are:

- Sensitive data is transmitted (via HTTP, FTP, SMTP, etc) or stored in clear-text (database, files, etc).
- Use of old or weak cryptographic algorithms.
- Use of weak or default encryption keys or re-use of compromised keys.

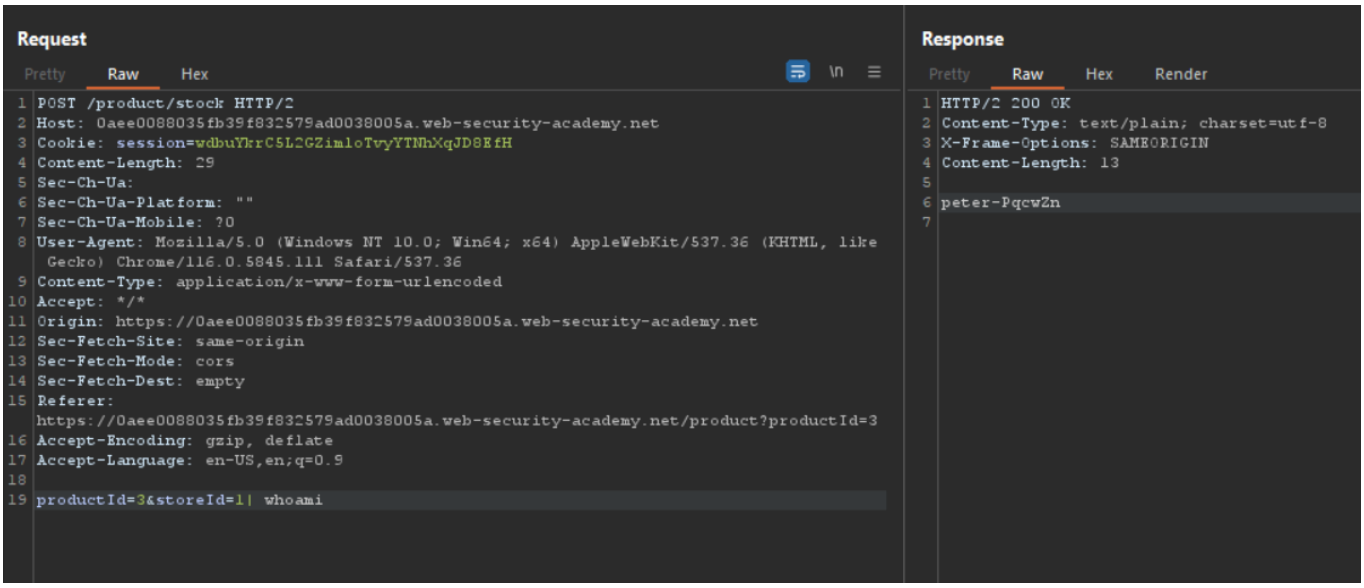
> No labs are available for this vulnerability to the best of my knowledge

A03:2021-Injection

slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.

Command injection

in this lab the the application is checking how many units of a product are remaining
we use the or character in bash to execute the whoami command to check who the current user is



A04:2021 – Insecure Design

Overview

A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include *CWE-209: Generation of Error Message Containing Sensitive Information*, *CWE-256: Unprotected Storage of Credentials*, *CWE-501: Trust Boundary Violation*, and *CWE-522: Insufficiently Protected Credentials*.

Example Attack Scenarios

Scenario #1: A credential recovery workflow might include “questions and answers,” which is prohibited by NIST 800-63b, the OWASP ASVS, and the OWASP Top 10. Questions and answers cannot be trusted as evidence of identity as more than one person can know the answers, which is why they are prohibited. Such code should be removed and replaced with a more secure design.

Scenario #2: A cinema chain allows group booking discounts and has a maximum of fifteen attendees before requiring a deposit. Attackers could threat model this flow and test if they could book six hundred seats and all cinemas at once in a few requests, causing a massive loss of income.

> No labs are available for this vulnerability to the best of my knowledge

A05:2021 – Security Misconfiguration

Moving up from #6 in the previous edition, 90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.%, and over 208k occurrences of a Common Weakness Enumeration (CWE) in this risk category. With more shifts into highly configurable software, it's not surprising to see this category move up. Notable CWEs included are *CWE-16 Configuration* and *CWE-611 Improper Restriction of XML External Entity Reference*.

Example Attack Scenarios

Scenario #1: The application server comes with sample applications not removed from the production server. These sample applications have known security flaws attackers use to compromise the server. Suppose one of these applications is the admin console, and default accounts weren't changed. In that case, the attacker logs in with default passwords and takes over.

Scenario #2: Directory listing is not disabled on the server. An attacker discovers they can simply list directories. The attacker finds and downloads the compiled Java classes, which they decompile and reverse engineer to view the code. The attacker then finds a severe access control flaw in the application.

> No labs are available for this vulnerability to the best of my knowledge

A06:2021 – Vulnerable and Outdated Components

It was #2 from the Top 10 community survey but also had enough data to make the Top 10 via data. Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used. Notable CWEs included are *CWE-1104: Use of Unmaintained Third-Party Components* and the two CWEs from Top 10 2013 and 2017.

You are likely vulnerable:

- If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
- If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs

and all components, runtime environments, and libraries.

- If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, leaving organizations open to days or months of unnecessary exposure to fixed vulnerabilities.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.
- If you do not secure the components' configurations (see A05:2021-Security Misconfiguration).

This vulnerability was demonstrated in the Mr. Robot ctf,
where you could get access to the cli of the server by exploiting a outdated wp cms and use the template functionality of the application to establish a reverse shell,
another vulnerability was that there were different responses to incorrect password attempt and incorrect username input which allowed us to easily brute force the admin login.

A07:2021 – Identification and Authentication Failures

Previously known as *Broken Authentication*, this category slid down from the second position and now includes Common Weakness Enumerations (CWEs) related to identification failures. Notable CWEs included are *CWE-297: Improper Validation of Certificate with Host Mismatch*, *CWE-287: Improper Authentication*, and *CWE-384: Session Fixation*.

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses if the application:

- Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
- Permits brute force or other automated attacks.
- Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
- Uses weak or ineffective credential recovery and forgot-password processes, such as "knowledge-based answers," which cannot be made safe.

- Uses plain text, encrypted, or weakly hashed passwords data stores (see A02:2021-Cryptographic Failures).
- Has missing or ineffective multi-factor authentication.
- Exposes session identifier in the URL.
- Reuse session identifier after successful login.
- Does not correctly invalidate Session IDs. User sessions or authentication tokens (mainly single sign-on (SSO) tokens) aren't properly invalidated during logout or a period of inactivity.

I solved the # Username enumeration via account lock lab in portswigger academy

here we have to use the cluster bomb functionality of burp suit and to get a response that says that the account has been locked due to too many attempts which tells us the username of the victim. Then we can brute-force the password as usual.

A08:2021 – Software and Data Integrity Failures

A new category for 2021 focuses on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data. Notable Common Weakness Enumerations (CWEs) include *CWE-829: Inclusion of Functionality from Untrusted Control Sphere*, *CWE-494: Download of Code Without Integrity Check*, and *CWE-502: Deserialization of Untrusted Data*.

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

> No labs are available for this vulnerability to the best of my knowledge

A09:2021 – Security Logging and Monitoring Failures

Security logging and monitoring came from the Top 10 community survey (#3), up slightly from the tenth position in the OWASP Top 10 2017. Logging and monitoring can be challenging to test, often involving interviews or asking if attacks were detected during a penetration test. There isn't much CVE/CVSS data for this category, but detecting and responding to breaches is critical. Still, it can be very impactful for accountability, visibility, incident alerting, and forensics. This category expands beyond *CWE-778 Insufficient Logging* to include *CWE-117 Improper Output Neutralization for Logs*, *CWE-223 Omission of Security-relevant Information*, and *CWE-532 Insertion of Sensitive Information into Log File*.

Returning to the OWASP Top 10 2021, this category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time:

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools (such as OWASP ZAP) do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

You are vulnerable to information leakage by making logging and alerting events visible to a user or an attacker (see A01:2021-Broken Access Control).

> No labs are available for this vulnerability to the best of my knowledge

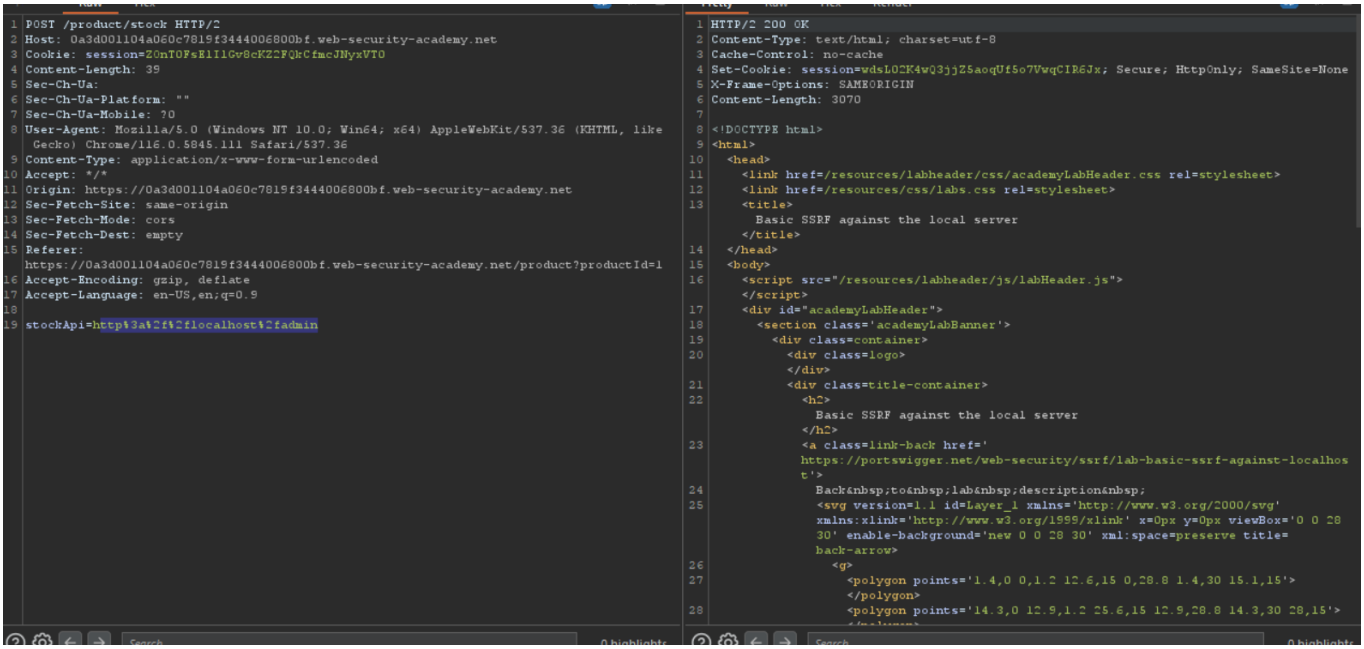
A10:2021 – Server-Side Request Forgery (SSRF)

This category is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage and above-average Exploit and Impact potential ratings. As new entries are likely to be a single or small cluster of Common Weakness Enumerations (CWEs) for attention and awareness, the hope is that they are subject to focus and can be rolled into a larger category in a future edition.

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.

the lab features a website where you can view products through an stockapi which redirects you to the next product using a path parameter every time you click next we can send request to get the admin panel by changing the path parameter to localhost:8080/admin which enables us to delete the admins users of the website



The grasshoppers are surprisingly easy to keep. They will keep your home free of bugs and vermin and need little else to eat. They are slightly jumpy abou being taken out on a leash, but with practice, you will find a way to fall in step quite quickly.

This particular breed has an exceptionally long lifespan and can be passed down through the generations. The grasshopper hasn't been cat, dog or child t so we highly recommend not having any visit your home. Can be housed with other grasshoppers, an older quiet one could help to show it the ropes and understand the rules of the house.

London Check stock

Web Security Academy

Basic SSRF against the local server

Back to lab description »

LAB Not solved

Home | Admin panel | My a

Users

wiener - Delete
carlos - Delete

< Retur